(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 103718155 A
(43)申请公布日 2014.04.09

(21)申请号 201180072813.6
(22)申请日 2011.10.08
(30)优先权数据
13/207806 2011.08.11 US
(85)PCT国际申请进入国家阶段日
2014.02.11
(86)PCT国际申请的申请数据
PCT/US2011/055492 2011.10.08
(87)PCT国际申请的公布数据
WO2013/022465 EN 2013.02.14
(71)申请人 微软公司
地址 美国华盛顿州
(72)发明人 B.E. 雷克托尔 E.H. 奥米亚
J.J. 杜尼茨 M.S. 洛弗尔
A. 霍尔塞克 M. 普拉克里亚
S.C. 罗 J.F. 斯普林菲尔德
N.R. 克罗斯 T.H. 巴苏
P.H. 杜苏德 R. 克里什纳斯瓦米
S.E. 卢科
(74)专利代理机构 中国专利代理(香港)有限公司 72001
代理人 曹芳 汪扬
(51)Int.Cl.
G06F 9/06(2006.01)
G06F 9/44(2006.01)

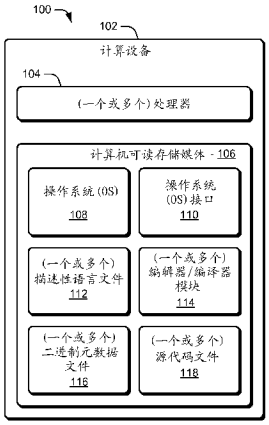
权利要求书1页 说明书11页 附图5页

(54)发明名称

运行时系统

(57)摘要

各种实施例提供了独立于编程语言来描述与操作系统相关联的一个或多个接口的能力。替换地或附加地,与特定编程语言相关联的编译器能够被配置成将(一个或多个)独立接口描述映射到所述特定编程语言。在一些实施例中,应用能够被配置成程序上确定所述操作系统的一个或多个接口。



CN 103718155 A

1. 一种计算机实现的方法，包括：

接收对与一个或多个可用操作系统接口相关联的信息的请求；

获得与所述一个或多个可用操作系统接口相关联的所述信息，所述操作系统接口使用抽象类型系统来描述；

确定与所述一个或多个可用操作系统接口相关联的一个或多个抽象类型；以及

将各个确定的抽象类型映射到与至少一个特定编程语言相关联的各个相应的类型。

2. 根据权利要求 1 所述的计算机方法，所述确定一个或多个抽象类型进一步包括确定与所述一个或多个抽象类型相关联的行为。

3. 根据权利要求 2 所述的计算机方法，所述确定与所述一个或多个抽象类型相关联的行为进一步包括通过至少一个二进制合同程序上确定行为。

4. 根据权利要求 1 所述的计算机方法，与所述一个或多个可用操作系统接口相关联的所述信息进一步包括层次信息。

5. 根据权利要求 1 所述的计算机方法，所述信息包括与至少一个接口相关联的版本信息、与所述至少一个接口相关联的各个参数的数据类型、以及与所述至少一个接口相关联的所述各个参数的调用顺序。

6. 根据权利要求 1 所述的计算机方法，所述获得与所述一个或多个可用操作系统接口相关联的信息包括从至少一个元数据文件程序上获得信息。

7. 一个或多个计算机可读存储媒体，包括计算机可读指令，所述计算机可读指令当被执行时，实现：

配置成使得能够实现对与操作系统相关联的功能性的程序访问的一个或多个应用编程接口 (API) 模块；

与所述一个或多个 API 模块相关联的一个或多个元数据文件，所述一个或多个元数据文件被配置成包括所述一个或多个 API 的至少一个描述，所述至少一个描述使用抽象类型系统；

配置成包括用于调用所述一个或多个 API 的一个或多个机器级别二进制合同的一个或多个应用二进制接口 (ABI) 模块；以及

配置成将至少一个类型的所述抽象类型系统映射到至少一个类型的一个或多个特定编程语言的一个或多个生成的语言投影模块。

8. 根据权利要求 7 所述的一个或多个计算机可读存储媒体，进一步包括当被执行时实现以下各项的指令：

配置成通过所述一个或多个元数据文件程序上确定在所述至少一个 API 模块中包括的至少一个接口的至少一个应用。

9. 根据权利要求 7 所述的一个或多个计算机可读存储媒体，所生成的语言投影模块包括至少一个代理，所述至少一个代理被配置成以本地于所述一个或多个特定编程语言的方式来仿真至少一个类型的所述抽象类型系统。

10. 根据权利要求 7 所述的一个或多个计算机可读存储媒体，用于调用所述一个或多个 API 的所述二进制合同包括与至少一个类型的所述抽象类型系统相关联的至少一个行为定义。

运行时系统

背景技术

[0001] 计算机现今经常地包括操作系统以管理对硬件和软件资源的访问。进而，操作系统能够包括各种类型的接口，诸如应用编程接口（API），以使得程序设计者和 / 或应用能够访问这些资源。在一些情况下，用记住的特定编程语言编程接口。当客户端应用被用特定编程语言编写时，客户端应用能够以本地于特定编程语言的方式来访问接口。然而，被用不同于接口的语言的语言编写的客户端应用有时可以利用附加的代码和 / 或包装器函数以便调用到接口中。

[0002] 典型地，接口被记入操作系统外部的文档。为了成功地调用接口以及编写包装器函数，程序设计者查阅外部文档资料以确定什么接口是可用的、什么是所关联的调用参数等。操作系统的新版本能够包括新的接口、对旧接口的修改以及废弃接口的去除。因此，为了维护当前一组 API，程序设计者不得不查阅关于操作系统的每个新版本的外部文档资料。这有时能够在当接口在操作系统中被释放时与当接口能够被包括在程序中时之间导致延迟。

发明内容

[0003] 本发明内容被提供来以简化的形式引入下文在具体实施方式中进一步描述的概念的选择。本发明内容不旨在标识所要求保护的主题的关键特征或基本特征，它也不旨在被用来限制所要求保护的主题的范围。

[0004] 各种实施例提供了独立于编程语言来描述与操作系统相关联的一个或多个接口的能力。替换地或附加地，与特定编程语言相关联的编译器能够被配置成将（一个或多个）独立接口描述映射到特定编程语言。在一些实施例中，应用能够被配置成程序上确定操作系统的一个或多个接口。

[0005] 一些实施例提供了利用抽象类型系统以面向对象的方式对操作系统的一个或多个接口建模的能力。在一些实施例中，扩展的接口定义语言（IDL）能够包括用来描述一个或多个接口之间的关系的语法。在一些实施例中，二进制合同能够被配置成暴露关联抽象类型系统的行为。

附图说明

[0006] 相同的标记在所有附图中被用来提及相似的特征。

[0007] 图 1 图示了依据一个或多个实施例的其中能够采用本文中所描述的各种原理的操作环境。

[0008] 图 2 图示了依据一个或多个实施例的架构。

[0009] 图 3 是描述依据一个或多个实施例的方法中的步骤的流程图。

[0010] 图 4 图示了依据一个或多个实施例的关系图。

[0011] 图 5 图示了能够被利用来实现一个或多个实施例的示例系统。

具体实施方式

[0012] 概述

各种实施例提供了独立于编程语言来描述与操作系统相关联的一个或多个接口的能力。抽象类型系统与接口定义语言（IDL）相组合地能够被配置成描述和/或定义接口。在一些实施例，能够以面向对象的方式来描述接口。替换地或附加地，与特定编程语言相关联的编译器能够被配置成将（一个或多个）语言无关的接口描述映射到在特定编程语言中的适当构造。替换地或附加地，编程语言应用运行时环境能够被配置成动态地（例如，在运行时执行期间）将（一个或多个）语言无关的接口描述映射到在特定运行时环境中的适当构造。在一些实施例中，应用能够被配置成程序上确定（一个或多个）独立接口描述。在一些情况下，能够以机器可读格式存储（一个或多个）独立接口描述。

[0013] 在以下讨论中，标题为“操作环境”的章节被提供并且描述了其中能够采用一个或多个实施例的一个环境。紧跟这个之后，标题为“操作系统组件访问”的章节描述了使得多个编程语言能够程序上访问系统组件的架构。接下来，标题为“用抽象类型系统对面向对象的语言建模”的章节描述了抽象类型系统如何能够与扩展的 IDL 相结合地被用来以面向对象的方式描述操作系统接口。最后，标题为“示例系统”的章节描述了能够被利用来实现一个或多个实施例的示例系统。

[0014] 已经提供了将在下面被描述的各种实施例的概述，现在考虑其中能够实现一个或多个实施例的示例操作环境。

[0015] 操作环境

图 1 一般地在 100 处图示依据一个或多个实施例的操作环境。环境 100 包括具有一个或多个处理器 104 和一个或多个计算机可读存储媒体 106 的计算设备 102。通过示例而非限制的方式，计算机可读存储媒体能够包括典型地与计算设备相关联的所有形式的易失性和非易失性存储器和/或存储媒体。这样的媒体能够包括 ROM、RAM、闪速存储器、硬盘、可移动媒体等等。计算设备的一个特定示例在下面在图 5 中被示出和描述。

[0016] 此外，计算设备 102 包括操作系统（OS） 108 和关联的（一个或多个）操作系统接口 110。虽然被示出为单独的模块，但应当领会和理解的是，操作系统 108 和（一个或多个）操作系统接口 110 能够被实现为单独的模块、组合的模块或其任何组合，而不背离所要求保护的主题的范围。操作系统 108 表示配置成管理计算设备 102 的（一个或多个）软件和/或硬件资源的功能性。（一个或多个）操作系统接口 110 表示对由操作系统 108 所提供的服务和/或功能性的程序访问，所述服务和/或功能性诸如存储器管理、文件管理、服务、函数、资源管理、外围设备管理等等。

[0017] 计算设备 102 同样包括一个或多个描述性语言文件 112，其表示配置成描述一个或多个接口的一个或多个文件。在一些实施例中，诸如（一个或多个）操作系统接口 110 之类的接口能够与操作系统相关联。描述性语言文件能够使用任何适合的描述、标记语言和/或语法（诸如用接口定义语言（IDL）、可扩展标记语言（XML）等等）来描述接口。

[0018] 此外，计算设备 102 同样包括一个或多个编辑器/编译器模块 114。在一些实施例中，编辑器/编译器模块 114 表示读取和/或解释（一个或多个）描述性语言文件 112 并且基于（一个或多个）文件 112 来生成诸如一个或多个二进制元数据文件 116 之类的输出的功能性。（一个或多个）二进制元数据文件 116 表示包括与（一个或多个）操作系统接

口 110 和 / 或操作系统 108 相关联的信息的一个或多个机器可读文件，所述信息诸如输入参数类型、参数调用顺序、接口之间的关系等等。

[0019] 替换地或附加地，（一个或多个）编辑器 / 编译器模块 114 表示读取和 / 或译解一个或多个源代码文件 118 的功能性。（一个或多个）源代码文件 118 对应于包含与至少一个编程语言相关联的编程语句的一个或多个文件。在一些情况下，（一个或多个）源代码文件 118 能够包括对一个或多个操作系统接口的调用的语句。（一个或多个）编辑器 / 编译器模块 114 解释（一个或多个）源代码文件 118，并且基于（一个或多个）源文件来生成机器可执行代码。

[0020] 计算设备 102 能够被体现为任何适合的计算设备，诸如（通过示例而非限制的方式）台式计算机、便携式计算机、笔记本计算机、诸如个人数字助理 (PDA)、手机之类的手持式计算机等等。

[0021] 已经描述了示例操作环境，现在考虑程序上将操作系统组件暴露给一个或多个编程语言的讨论。

[0022] 操作系统组件访问

在计算设备上运行的应用经常地利用由在计算设备上运行的操作系统所提供的特征。操作系统能够使得实现对计算设备上的关联资源的简化访问，以及提供服务。有时，能够程序上访问这些特征、服务和 / 或资源。然而，如果操作系统以不同于应用所被编写的编程语言的编程语言格式来暴露这些功能性，则程序设计者将典型地编写包装器函数以帮助不同编程语言之间的转变。例如，考虑已被编写和 / 或暴露为扁平、导出的“C”函数的接口。期望使用“C”风格函数的 C# 或 Visual Basic 程序设计者能够包括特殊语句和 / 或附加的代码，以便使得它们的编程语言能够成功地调用“C”风格函数。结果，用与所暴露的接口不同的编程语言编写的应用不能够访问接口直到附加的语句和 / 或包装器函数被编写了为止。

[0023] 各种实施例提供了独立于编程语言来描述与操作系统相关联的一个或多个接口的能力。独立于编程语言来描述接口使得多个编程语言能够容易地访问接口。当编程语言学习到如何解释接口的语言无关描述时，这个知识能够被应用于使用语言无关描述的现有和将来的接口。在一些实施例中，能够使用抽象类型系统来描述接口。进而，特定编程语言或关联的编程语言编译器能够将抽象类型系统映射到它关联的类型系统以解释和 / 或访问接口。

[0024] 考虑图 2，图 2 图示了依据一个或多个实施例的架构 200。架构 200 包括操作系统 202，所述操作系统 202 能够被配置成在计算设备上执行。应该理解的是为了简洁起见，操作系统 202 未被整体地图示。操作系统 202 包括配置成管理与计算设备相关联的资源的一个或多个操作系统组件 204。在一些实施例中，（一个或多个）操作系统组件 204 能够提供对资源以及与管理资源相关联的一个或多个服务和 / 或特征的程序访问。（一个或多个）操作系统组件 204 还能够包括与操作系统 202 相关联的基本元素以及从基本元素建立的复杂元素。

[0025] 在一些实施例中，能够经由诸如 API 之类的一个或多个接口来暴露（一个或多个）操作系统组件 204。在这个示例中，操作系统 202 包括新的 API 族 206、基于 COM 的 API 族 208 以及基于扁平导出的 API 族 210。新的 API 族 206 表示一个或多个相关的 API，其中功能性（即类、接口、方法、属性、事件等等）直接地使用抽象类型系统来描述，如在下面进一

步描述的那样。基于 COM 的 API 族 208 表示其中功能性使用组件对象模型 (COM) 类型系统来描述的一个或多个 API。基于扁平导出的 API 族 210 表示其中功能性使用方法签名（即包括方法名称、调用约定、方法变元的数目和类型的方法签名）来描述的一个或多个 API。基于扁平导出的 API 进一步表示仅由它们的名称标识并且没有被布置成类和 / 或面向对象的方式的 API。想要确定哪些 API 是可用的程序设计者能够手动地和 / 或程序上访问每个 API 的描述。例如，为了确定对于新的 API 族 206 存在什么接口并且如何调用它们，程序设计者能够访问关联的元数据 212。虽然基于 COM 的 API 族 208 和基于扁平导出的 API 族 210 分别在元数据 214 和 216 中具有关联的语言无关类型系统描述，但是程序设计者首先编写包装器代码以将语言无关类型系统描述映射到基于 COM 的和 / 或基于扁平导出的 API。

[0026] 元数据 212、214 以及 216 能够被配置成包括描述关联的（一个或多个）接口的各种方面的信息，诸如版本信息、什么方法是可用的、（一个或多个）接口采取什么参数、参数的数据类型、传递参数的顺序等。在一些实施例中，元数据能够包括与接口相关联的层次 (hierarchical) 信息，诸如描述（一个或多个）接口之间的关系和 / 或以面向对象的方式来描述（一个或多个）接口的信息。元数据能够被配置成包括类描述、关联的方法以及类的参数等等。在一些情况下，一个或多个 IDL 文件能够被扩展成包括这些描述中的一些并且被用在一个或多个元数据文件的生成中。在一些情况下，元数据可以是至少部分地基于一个或多个 IDL 文件的，如在下面进一步描述的那样。

[0027] 操作系统 202 同样包括（一个或多个）应用二进制接口 (ABI) 218。ABI 在机器级别描述用于调用函数、方法、API 等等的二进制合同。二进制合同能够包括与函数相关联的标识或名称、能够被用来调用函数的签名、传递给函数的参数的顺序和 / 或与参数相关联的数据类型等。替换地或附加地，二进制合同能够包括用于暴露与至少一个类型的类型系统相关联的行为的定义和 / 或规则。典型地，与二进制合同相关联的和 / 或由二进制合同所定义的行为不改变。例如，如果二进制合同的签名和 / 或标识保持不变，则所关联的合同的行为也保持不变。

[0028] 应用二进制接口 218 表示通过能够被其他应用可靠地援引的二进制暴露的功能性。在这个示例中，（一个或多个）应用二进制接口 218 包括与（一个或多个）操作系统组件 204 相关联的接口、基础类型以及基础模式。在操作系统 202 外部的一个或多个应用（诸如（一个或多个）应用 220）能够经由一个或多个应用二进制接口 218 来访问操作系统组件 204。

[0029] 应用 220 能够包括从诸如 HTML、JavaScript、Visual Basic、C#、C++ 等等之类的一个或多个编程语言生成的一个或多个应用。在一些实施例中，应用 220 将一个或多个调用包括到操作系统组件中。在一些情况下，（一个或多个）应用 220 能够被配置成首先程序上确定什么接口是可用的，并且然后做出到所确定的（一个或多个）接口中的一个或多个的调用。在一些情况下（一个或多个）应用 220 在来自如在下面进一步描述的一个或多个生成的语言投影模块 222 的帮助下通过（一个或多个）应用二进制接口 218 来访问（一个或多个）接口。

[0030] 在一个或多个实施例中，生成的（一个或多个）语言投影模块 222 将抽象类型系统定义映射到特定编程语言。能够映射任何适合的编程语言，其示例在上面被提供。在一些实施例中，生成的语言投影模块对于每个编程语言来说可以是唯一的。在其他实施例中，

生成的语言投影模块可以是多目的的并且被多个编程语言利用。映射使得能够实现使用抽象类型系统来描述成在没有附加的编程语句（即包装器函数）的情况下对于特定编程语言可访问的当前的和将来的接口。映射进一步允许特定编程语言以本地于特定编程语言的方式来调用接口。能够映射任何适合类型的信息，诸如类、数据类型、函数指针、结构等等。

[0031] 考虑其中程序设计者想要访问操作系统组件的示例。当编写诸如（一个或多个）应用 220 之类的应用时，程序设计者在与至少一个特定编程语言相关联的编辑器 / 编译器中生成源代码。编辑器 / 编译器能够被配置成访问（一个或多个）元数据文件以便获得和与操作系统相关联的什么接口和 / 或 API 是可用的相关联的信息。例如，在一些实施例中，当程序设计者编写了包括对作为操作系统组件的一部分实现的类的调用的一行代码时，编译器 / 编辑器能够查询与类相关联的元数据并且将什么方法、属性等等与类相关联的列表返回给程序设计者。列表能够包括任一种信息、关联的方法和 / 或类的属性等等。替换地或附加地，列表能够包括可用类的列表。在一些实施例中，信息能够被提供为配置成在视觉上将接口的有关方法、属性等呈现给用户以供选择的自动完成特征的一部分。在选择方法和 / 或属性时，编译器 / 编辑器能够将对应的语法插入到源代码中以得到更高效且准确的源代码生成。

[0032] 在一些实施例中，程序设计者可以编写配置成使操作系统组件的类对象实例化的源代码。当在运行时被调用时，操作系统动态地创建类的实例以交付给调用代码。然而，交付给调用代码的实例可以是“抽象对象”或在与操作系统组件相关联的抽象类型系统中所描述的对象，如在下面进一步描述的那样。为了在抽象对象类型和与调用代码相关联的编程语言的具体数据类型之间桥接，编译器能够被配置成用它关联的编程语言（诸如通过生成的（一个或多个）语言投影模块 222）将抽象对象转化和 / 或映射到可比较的类型。在一些情况下，代理能够被用来桥接操作系统组件的抽象对象和与编程语言相关联的具体对象之间的调用。

[0033] 考虑配置成仿真操作系统组件类的行为的代理。在一些情况下，能够创建包括类的关联类型、方法、属性、事件、接口等的占位程序（stub）的代理。能够用调用代码的编程语言来构造代理，从而使得代码调用能够以本地于调用代码的方式来访问代理。占位程序能够包括适当的知识和 / 或代码以将这些调用转化和 / 或映射到操作系统（并且从操作系统转化和 / 或映射）。例如，在一些实施例中，代理能够与（一个或多个）应用二进制接口 218 通信。

[0034] 在一些实施例中，编程语言能够像上面所描述的那样插进包装器函数和 / 或代理，以将抽象类型映射到本地于编程语言的类型。编程语言编辑器 / 编译器能够被配置成读取元数据（诸如元数据文件 212、214 以及 216），确定什么抽象类型正被使用，用所关联的程序语言将（一个或多个）抽象类型映射到一个或多个可比较的类型，并且将关联的包装器函数和 / 或代理接线到与操作系统组件相关联的抽象类型。一旦用于每个类型的映射存在于编程语言与抽象类型系统之间，由抽象类型系统所定义的任何当前的或将来的接口就能够自动地被编程语言访问，而不用来自程序设计者的附加的编码。

[0035] 作为示例，考虑图 3，图 3 图示了描述依据一个或多个实施例的方法中的步骤的流程图。该方法能够被任何适合的硬件、软件、固件或其组合实行。在至少一些实施例中，该方法的方面被诸如在计算设备 102 上执行的编辑器 / 编译器模块 114 之类的软件实行。

[0036] 步骤 302 接收对与可用操作系统接口相关联的信息的请求。例如，该请求能够被配置成请求什么接口是可用的和 / 或与可用接口相关联的其他信息。能够以任何适合的方式实现这一点。在一些实施例中，能够由其中源代码正被开发的源代码编辑器自动地生成请求。代码编辑器能够标识对操作系统接口和 / 或组件的调用，并且在标识时，发送对与操作系统接口相关联的可用方法、属性等的请求。在一些实施例中，能够经由下拉菜单、单选按钮等等的选择手动地生成请求。请求能够被配置成请求与所有可用操作系统接口、一些可用操作系统接口、各个操作系统接口或其组合相关联的信息。在一些实施例中，能够经由诸如图 2 的（一个或多个）应用 220 之类的运行的应用来生成请求。

[0037] 步骤 304 获得与一个或多个操作系统接口相关联的信息。例如，在一些实施例中，操作系统能够被查询以得到信息。替换地或附加地，信息能够被包括在一个或多个元数据文件中并且通过查询和 / 或读取元数据文件来获得。如上面和在下面所描述的那样，元数据文件能够以机器可读格式，并且包括层次信息，诸如与接口相关联的对象关系构成。

[0038] 响应于获得信息，步骤 306 确定与一个或多个操作系统接口相关联的一个或多个抽象类型。如上面所描述的那样，在一些实施例中，能够通过利用抽象类型系统独立于特定编程语言来描述操作系统接口。响应于确定了与一个或多个操作系统接口相关联的一个或多个抽象类型，步骤 310 将一个或多个抽象类型中的每一个映射到与特定编程语言相关联的类型。

[0039] 考虑将 ABI 描述为 OpenPicker 运行时类的接口描述语言的示例。被包括在 OpenPicker 运行时类中的是配置成返回 FileItem 对象的合集的 PickMultipleItems 方法。在这个特别的示例中，FileItem 类包含配置成返回包括 Name（名称）值的 FileItemProperties 对象的 GetProperties 方法。

```
runtimeclass OpenPicker {
    interface IOpenPicker;
}
interface IOpenPicker : IInspectable {
    HRESULT PickMultipleItems( [out, retval] IVector<IFileItem*>* folder );
}
interface IFileItem : IInspectable {
    HRESULT GetProperties([out, retval] FileItemProperties** retVal);
}
interface IFileItemProperties : IInspectable {
    [propget] HRESULT Name([out, retval] HSTRING *value);
}
```

[0040] 在一些实施例中，C# 语言编译器能够将描述映射到 C# 语言构造。通过将描述映射到语言特定的构造，能够以本地于 C# 编程语言的方式来访问操作系统接口，如在下面所说明的那样：

```
OpenPicker picker = new OpenPicker();
var items = picker.PickMultipleItems();
foreach (FileItem item in items) {
   Display (item.GetProperties().Name);
}
```

在又一个示例中，以语言无关的方式所定义的结构能够被映射到 JavaScript 对象。考虑使用下项定义的 FrameRate 结构的示例：

```
typedef struct FrameRate {
 UINT32 Numerator;
 UINT32 Denominator;
} FrameRate;
```

FrameRate 结构包括两个 UINT32 字段：Numerator 和 Denominator。因为 FrameRate 结构使用语言无关项来定义，所以该结构能够被各种语言特定的编程在使用适当的映射时访问，诸如通过 JavaScript 被访问。然而，JavaScript 不包括包含字段的结构的概念或无符号 32 位整数的概念。替代地，JavaScript 包括具有属性的对象的概念和 Number 的概念。在这个示例中，上述结构定义可以是具有命名为 Numerator 和 Denominator 的两个属性的对象，两者类型为 Number：

```
// JavaScript use of the above
var framerate = videoType.Framerate; // Get a framerate object
var ratio = framerate.Numerator / framerate.Denominator;
```

已经考虑了程序上将操作系统组件暴露给一个或多个编程语言，现在考虑依据一个或多个实施例用抽象类型系统对面向对象的语言建模的讨论。

[0041] 用抽象类型系统对面向对象的语言建模

当操作系统接口已使用特定编程语言来定义时，与接口相关联的参数由特定编程语言的数据类型来表征。进而，接口的调用者符合特定编程语言如何定义数据类型。对于用特定编程语言编写为为接口的程序，符合数据类型变得无关紧要，因为该程序具有相同的数据类型定义。调用该接口的不用特定编程语言编写的程序有时可能降低它们的能力以便符合数据类型。例如，调用来自不同编程语言的接口的程序设计者可以添加代码以在不同的编程语言和 / 或数据类型之间转换和 / 或桥接。结果，数据类型之间的桥接过程经常地能够降低调用编程语言的数据类型固有的功能性。附加地，这个转换有时可能是复杂的，并且在一些情况下，如果不同的编程语言具有对象是什么的不同概念则转换是不存在的。

[0042] 各种实施例提供了利用抽象类型系统以面向对象的方式对操作系统的一个或多个接口建模的能力。在一些实施例中，扩展的 IDL 能够包括用来描述一个或多个接口之间的关系的语法。替换地或附加地，扩展的 IDL 能够包括支配接口如何能够被描述和 / 或定义的规则。在一些实施例中，二进制合同能够被配置成暴露关联抽象类型系统的行为，并且

进一步被配置成包括与聚合数据类型相关联的信息。

[0043] 面向对象的模型使用类层次来描述对象，其中类能够包括方法、属性、继承等。典型地，但未必，面向对象的类包括一个或多个构造器。构造器是类函数，所述类函数当被调用时，在存储器中创建类的实例 / 对象以供使用。类的一些构造器包含准备所创建的实例 / 对象以供使用的附加的代码。例如，附加的代码能够初始化类的成员变量，运行类的关联的启动例程等。默认构造器没有输入参数，而非默认构造器具有一个或多个输入参数。

[0044] 在一些实施例中，操作系统组件的一个或多个构造器能够被约束成遵循特别的实施方式和 / 或设计。通过遵循特别的实施方式，构造器的调用者能够取决于构造器和 / 或作为结果的对象如何表现。例如，操作系统组件的默认构造器能够被约束和 / 或映射到类工厂对象的 IActivationFactory 接口的 ActivateInstance 方法的默认构造器。类工厂对象是建模来创建对象而不用指定所创建对象的确切类的设计模式。通过使用类工厂模型，默认构造器能够维持充足水平的抽象。操作系统组件的非默认构造器同样能够被约束成被放置在一个或多个特殊地指定的接口（诸如类工厂的可替换的接口）上。

[0045] 指定接口的描述的一个方式是通过使用描述性语言文件，诸如像 IDL 和 / 或 XML 这样的扩展描述语言。在一些实施例中，扩展描述语言能够被配置成使得能够实现用来描述和 / 或指定操作系统组件的一个或多个构造器接口的能力。例如，考虑 Compressor（压缩器）类，如在下面所说明的那样。在这个示例中，Compressor 类声明它的 ICompressorFactory 接口包含与该类相关联的（一个或多个）非默认构造器方法的至少一个定义。

```
[version(NTDDI_WIN8), activatable(ICompressorFactory, NTDDI_WIN8)]
runtimeclass Compressor {
    [default] interface ICompressor;
}
[version(NTDDI_WIN8), uuid(5F3D96A4-2CFB-442C-A8BA-D7D11B039DA0)]
interface ICompressorFactory : IInspectable {
    HRESULT CreateCompressor([in] Windows.Foundation.IOutputStream *
 UnderlyingStream, [in] CompressAlgorithm Algorithm, [out, retval] Compressor
 **CreatedCompressor);
 }
```

[0046] 这些描述能够被包含在一个或多个文件中并且与至少一个编译器相结合地用来以机器可读格式生成一个或多个接口描述，如在下面进一步描述的那样。

[0047] 面向对象的类能够被配置成包括至少一个静态方法、至少一个静态属性、至少一个静态事件或其任何组合。与约束构造器的定义类似，静态方法、静态属性以及和 / 或静态事件能够被设计成符合一个或多个特定地指明的接口。考虑声明静态方法、属性以及事件的 CallControl 类的示例。在这个示例中，静态方法、属性以及事件被定义在 ICallControlStatics 接口中：

```
[version(NTDDI_WIN8), uuid("03945AD5-85AB-40E1-AF19-56C94303B019"),
exclusiveto(CallControl)]
    interface ICallControlStatics: IInspectable
    {
      HRESULT GetDefault([out, retval] CallControl **callControl);
      HRESULT FromId([in] HSTRING deviceInterfaceId, [out, retval] CallControl
**callControl);
    }
                    // Runtime classes
                    [version(NTDDI_WIN8),
                     static(ICallControlStatics, NTDDI_WIN8)]
                    runtimeclass CallControl{
                        [default] interface ICallControl;
                    }
```

面向对象的类能够被配置成包括至少一个实例方法、至少一个实例属性、至少一个实例事件或其任何组合。实例成员（方法、属性以及事件）对类的指定实例进行操作而静态成员被类的所有成员共享。类能够被设计成符合一个或多个特定地指定的接口，诸如其中CallControl类声明它关联的实例方法、属性以及事件被定义在ICallControl接口中的上述示例。在一些实施例中，一个或多个操作系统组件能够在类工厂设计模式之后/上对这些接口建模/实现这些接口。

[0048] 无论特定编程语言是否调用语言特定的构造器、静态或实例成员，语言投影模块（诸如图2的生成的语言投影模块222）都能够将编程语言特定的调用映射到与适当的类工厂对象或实例相关联的适当方法和/或接口。在一些实施例中，类工厂对象与操作系统组件和/或运行时类相关联。例如，指导编译器创建运行时类的实例的代码的方法签名能够与类工厂对象的适当的构造器接口方法相匹配。在又一个示例中，语言投影模块能够被配置成使指导编译器调用静态方法的语言特定的方法签名与在与操作系统组件相关联的静态接口上的适当匹配接口相匹配。在又一个示例中，语言投影模块能够被配置成使指导编译器调用实例方法的语言特定的方法签名与在与操作系统组件的特别的实例相关联的接口上的适当匹配接口方法相匹配。

[0049] 在一些实施例中，IDL编译器能够被配置成读取与至少一个操作系统组件接口描述相关联的一个或多个扩展的IDL文件，并且生成关联的机器可读文件。任何适合类型的信息能够被包括在扩展的IDL文件中，所述信息诸如描述关联的操作系统组件的类型、方法、属性、事件以及接口的信息。在一些情况下，编译器能够生成机器可读元数据文件。一个或多个应用能够被配置成读取所关联的机器可读文件以程序上确定什么接口是可用的、关联的描述和/或接口的声明、接口的参数的数据类型等。能够用任何适合的编程语言来编写一个或多个应用，所述任何适合的编程语言的示例在上面被提供。

[0050] 在一些实施例中，IDL编译器能够生成与类型、方法、属性、事件和/或接口相关联

的一个或多个通信代理和 / 或(一个或多个)代码占位程序。通信代理和 / 或代码占位程序能够附加地被配置成用于由操作系统组件访问。

[0051] 考虑图 4,图 4 图示了依据一个或多个实施例的(一个或多个)扩展的 IDL 文件 402、编译器 404 与(一个或多个)元数据文件 406 之间的关系。在这里,(一个或多个)扩展的 IDL 文件 402 被编译器 404 接收和处理以产生(一个或多个)元数据文件 406。在至少一些实施例中,在关系中所图示的模块能够被实现为软件、硬件或其任何组合,诸如在计算设备 102 上执行的编辑器 / 编译器模块 114。

[0052] 在所图示和描述的实施例中,(一个或多个)扩展的 IDL 文件 402 能够包括定义一个或多个操作系统组件接口(诸如操作系统 API)的一个或多个文件。能够描述任何适合类型的操作系统组件,诸如文件对象 / 类、串对象 / 类、图形对象 / 类、文化对象 / 类等等。每个对象能够包括能够与面向对象的类相关联的方法、属性、继承信息等。扩展的 IDL 能够包括使得能够实现要描述的一个或多个接口之间的这些类型的关系的语法。替换地或附加地,扩展的 IDL 能够描述抽象类型系统。

[0053] 在一些实施例中,抽象类型系统能够被配置成支持和 / 或描述各种数据类型。例如,方法接口能够被定义来映射所有接口对对象的实例进行操作的物理要求。在另一示例中,抽象类型系统能够被用来定义能够跨越编程语言和组件边界被调用的函数指针。第一组件能够发起函数指针或被绑定到方法的代表。然后代表能够被跨越二进制编译边界传递到用任意编程语言编写的第二组件,并且被援引。各种实施例使得扩展的 IDL 能够定义和 / 或描述这些数据类型,以便定义用于组成基本数据类型以及将该数据类型复合成聚合类型的规则。

[0054] 在所图示和描述的实施例中,编译器 404 接受一个或多个输入文件(诸如扩展的 IDL 文件 402),并且产生一个或多个元数据文件 406。(一个或多个)元数据文件 406 能够被配置用于自动化的访问。例如,能够以机器可读格式存储元数据文件。在一些实施例中,(一个或多个)元数据文件 406 与一个或多个操作系统组件相关联。替换地或附加地,编译器 404 能够生成被占位的通信代理,如上面所描述的那样。

[0055] 诸如通过读取(一个或多个)元数据文件 406,(一个或多个)应用 408 能够动态地确定什么 API 是可用的。在一些实施例中,(一个或多个)应用 408 能够被配置为图 1 的编辑器 / 编译器 114。通过(一个或多个)元数据文件 406,应用能够确定是否存在功能性的较近版本、API 采取什么参数、以及向用户通知用于 API 的什么功能性在运行时存在。因此,通过以机器可读格式来包括 API 描述以及用抽象类型系统来描述 API,支持抽象类型系统与特定编程语言之间的映射的应用和 / 或语言用来自程序设计者的较少努力就容易地访问 API。

[0056] 已经考虑了用抽象类型系统对面向对象的语言建模,现在考虑依据一个或多个实施例的示例系统的讨论。

[0057] 示例系统

图 5 图示了能够被用来实现上面所描述的各种实施例的示例计算设备 500。计算设备 500 可以是例如图 1 的计算设备 102 或任何其他适合的计算设备。

[0058] 计算设备 500 包括一个或多个处理器或处理单元 502、一个或多个存储器和 / 或存储组件 504、一个或多个输入 / 输出 (I/O) 设备 506 以及允许各种组件和设备与彼此通信的

总线 508。总线 508 表示一个或多个数个类型的总线结构中的任一个，包括存储器总线或存储器控制器、外围总线、加速图形端口、以及使用各种总线架构中的任一个的处理器或局部总线。总线 508 能够包括有线和 / 或无线总线。

[0059] 存储器 / 存储组件 504 表示一个或多个计算机存储媒体。组件 504 能够包括易失性媒体（诸如随机存取存储器 (RAM)）和 / 或非易失性媒体（诸如只读存储器 (ROM)、闪速存储器、光盘、磁盘等等）。组件 504 能够包括固定媒体（例如，RAM、ROM、固定硬盘驱动器等）以及可移动媒体（例如，闪速存储器驱动器、可移动硬盘驱动器、光盘等等）。

[0060] 一个或多个输入 / 输出设备 506 允许用户将命令和信息键入到计算设备 500，并且同样允许信息被呈现给用户和 / 或其他组件或设备。输入设备的示例包括键盘、光标控制设备（例如，鼠标）、麦克风、扫描器等等。输出设备的示例包括显示设备（例如，监视器或投影仪）、扬声器、打印机、网络卡等等。

[0061] 可以在本文中在软件或程序模块的一般上下文中描述各种技术。一般地，软件包括执行特定任务或者实现特定抽象数据类型的例程、程序、对象、组件、数据结构等等。可以在某种形式的计算机可读媒体上存储或者跨越某种形式的计算机可读媒体传输这些模块和技术的实施方式。计算机可读媒体可以是能够被计算设备访问的任何可用的介质或媒体。通过示例而非限制的方式，计算机可读媒体可以包括“计算机可读存储媒体”。

[0062] “计算机可读存储媒体”包括用任何方法或技术实现以用于诸如计算机可读指令、数据结构、程序模块或其他数据之类的信息的存储的易失性和非易失性、可移动和非可移动媒体。计算机可读存储媒体包括但不限于 RAM、ROM、EEPROM、闪速存储器或其他存储器技术、CD-ROM、数字通用盘 (DVD) 或其他光学存储装置、磁盒、磁带、磁盘存储装置或其他磁存储设备、或能够被用来存储所期望的信息并且能够被计算机访问的任何其他介质。

[0063] 结论

各种实施例提供了独立于编程语言来描述与操作系统相关联的一个或多个接口的能力。替换地或附加地，与特定编程语言相关联的编译器能够被配置成将（一个或多个）独立接口描述映射到特定于编程语言的适当构造。在一些实施例中，应用能够被配置成程序上确定操作系统的一个或多个接口。

[0064] 尽管已经用特定于结构特征和 / 或方法学行为的语言描述了本主题，但应当理解的是，在所附权利要求中限定的本发明的主题未必限于上面所描述的特定特征或行为。相反地，上面所描述的特定特征和行为作为实现权利要求的示例形式被公开。

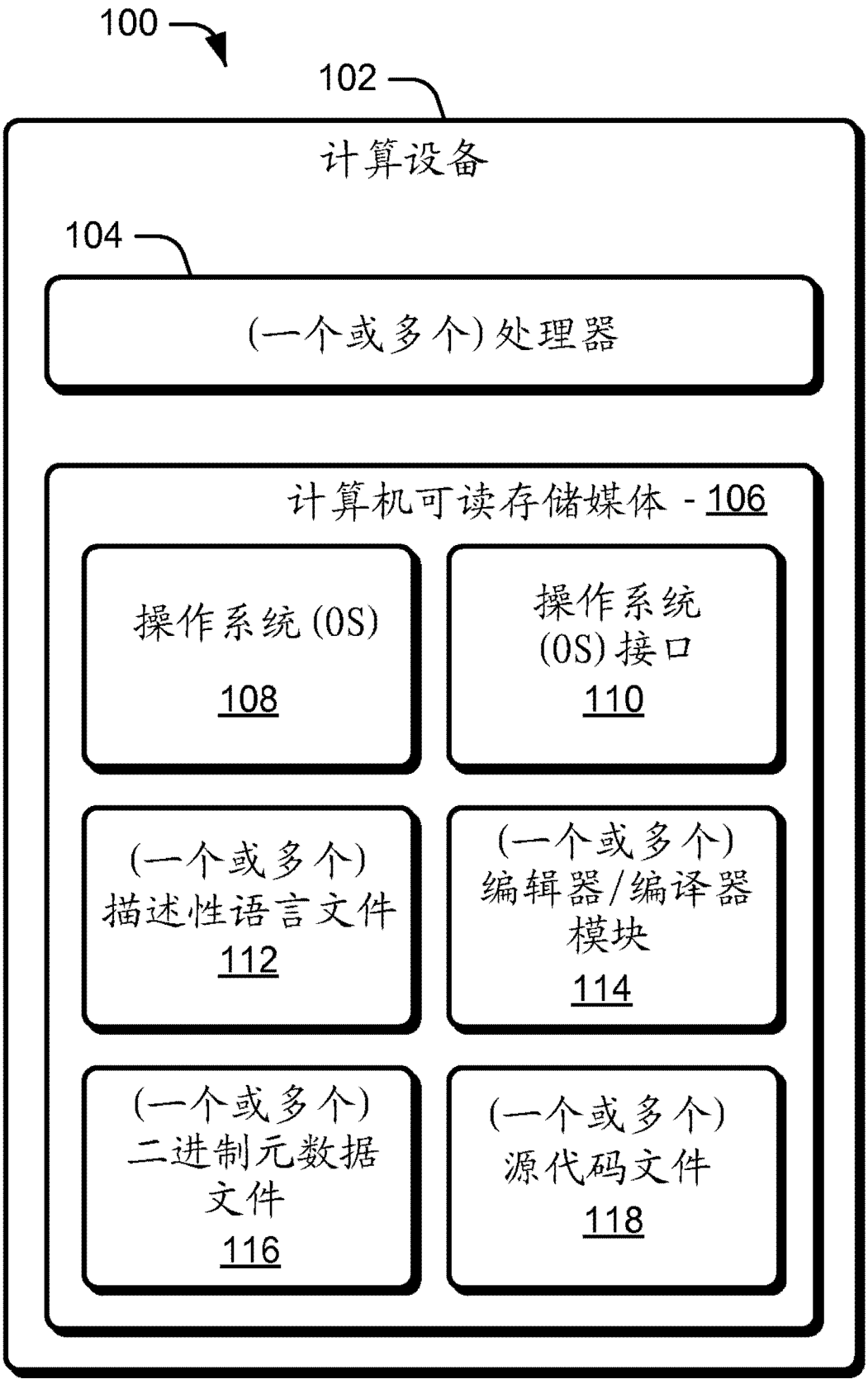

100
102
计算设备
104
(一个或多个)处理器
计算机可读存储媒体 - 106
操作系统(OS)
108
操作系统
(OS)接口
110
(一个或多个)
描述性语言文件
112
(一个或多个)
编辑器/编译器
模块
114
(一个或多个)
二进制元数据
文件
116
(一个或多个)
源代码文件
118

图 1

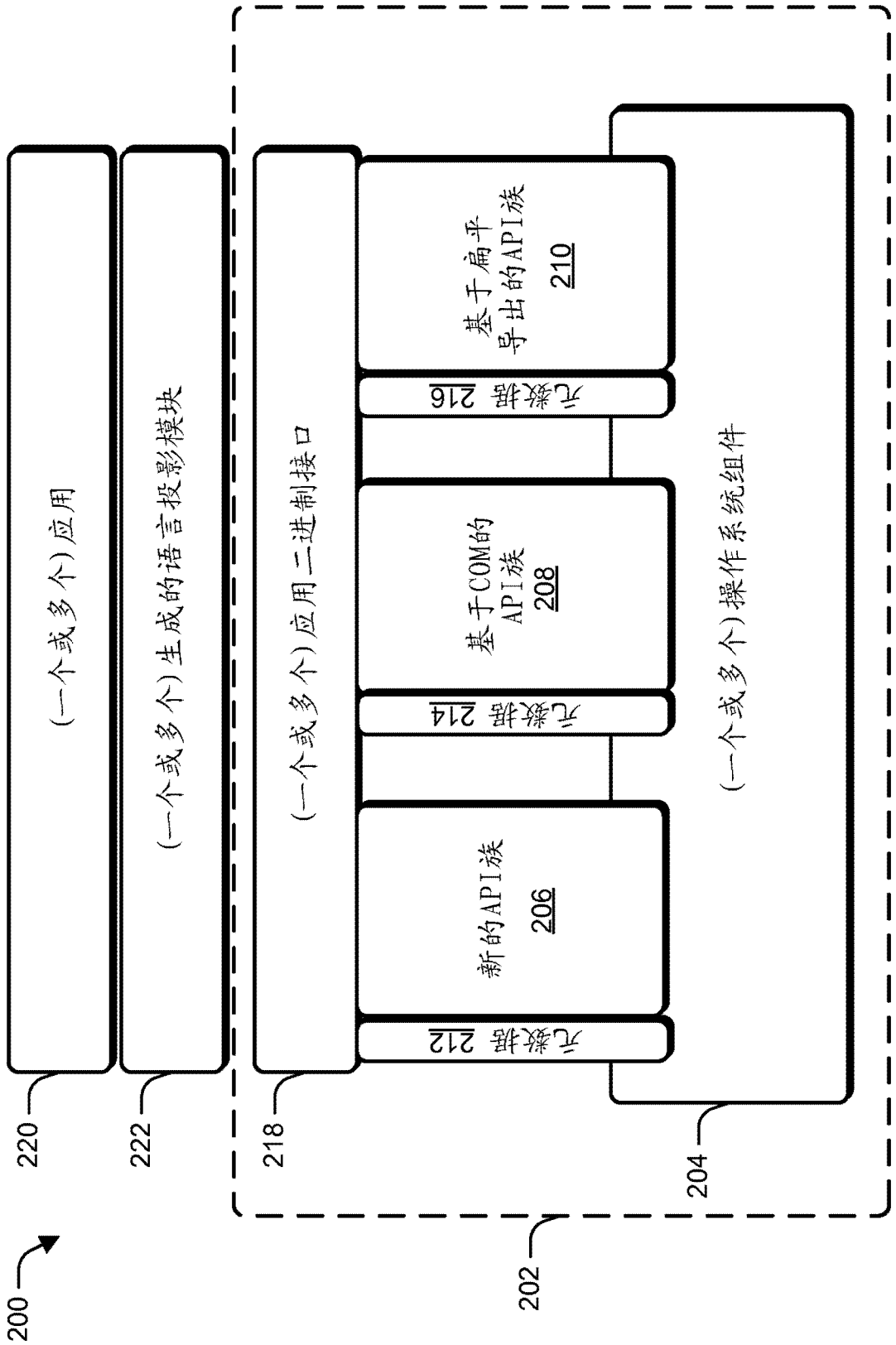

(一个或多个)应用
(一个或多个)生成的语言投影模块
(一个或多个)应用二进制接口
基于扁平导出的API族 210
元数据 216
基于COM的API族 208
元数据 214
新的API族 206
元数据 212
(一个或多个)操作系统组件
200
202
204
218
220
222

图 2

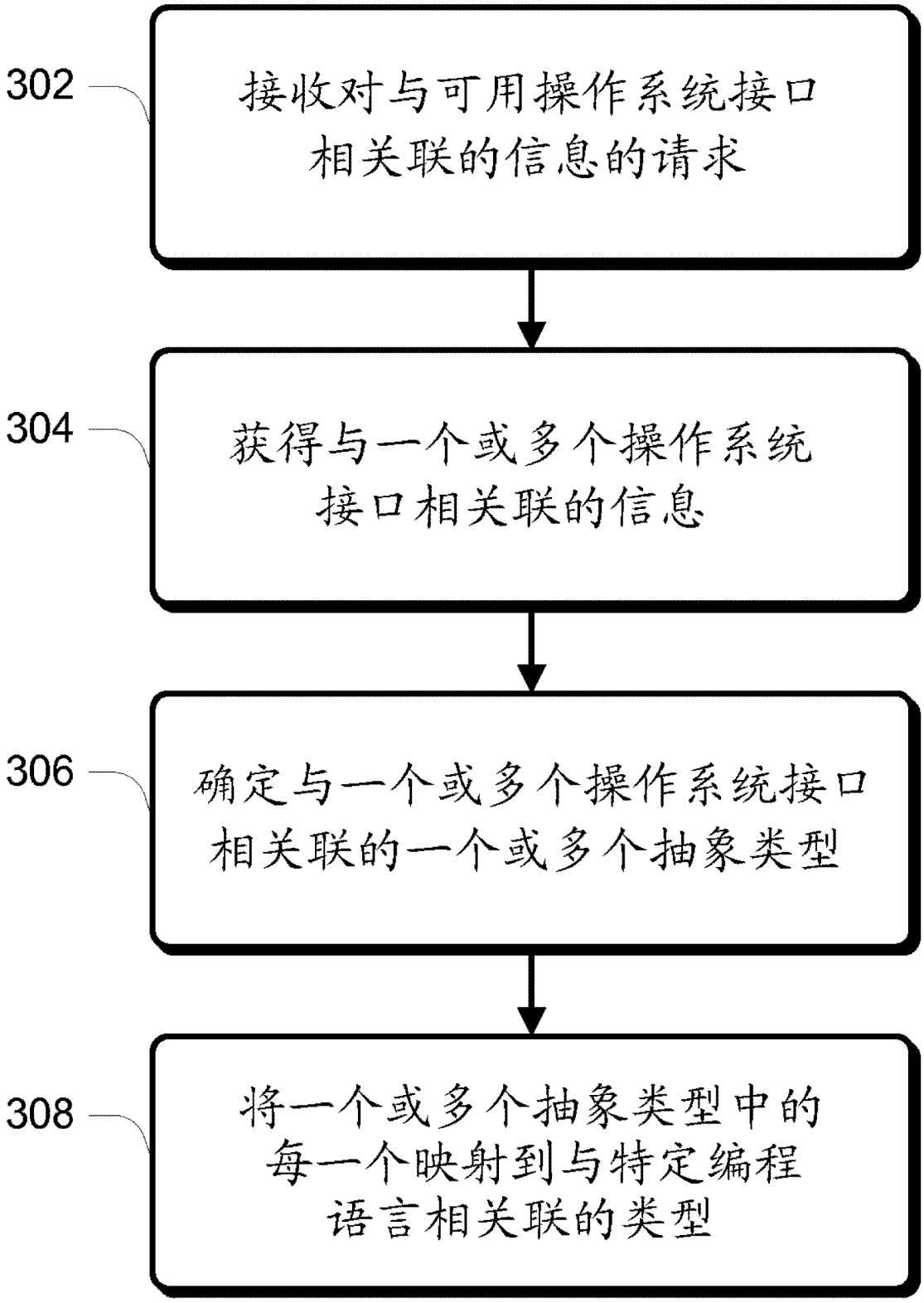

302
接收对与可用操作系统接口相关联的信息的请求
304
获得与一个或多个操作系统接口相关联的信息
306
确定与一个或多个操作系统接口相关联的一个或多个抽象类型
308
将一个或多个抽象类型中的每一个映射到与特定编程语言相关联的类型

图 3

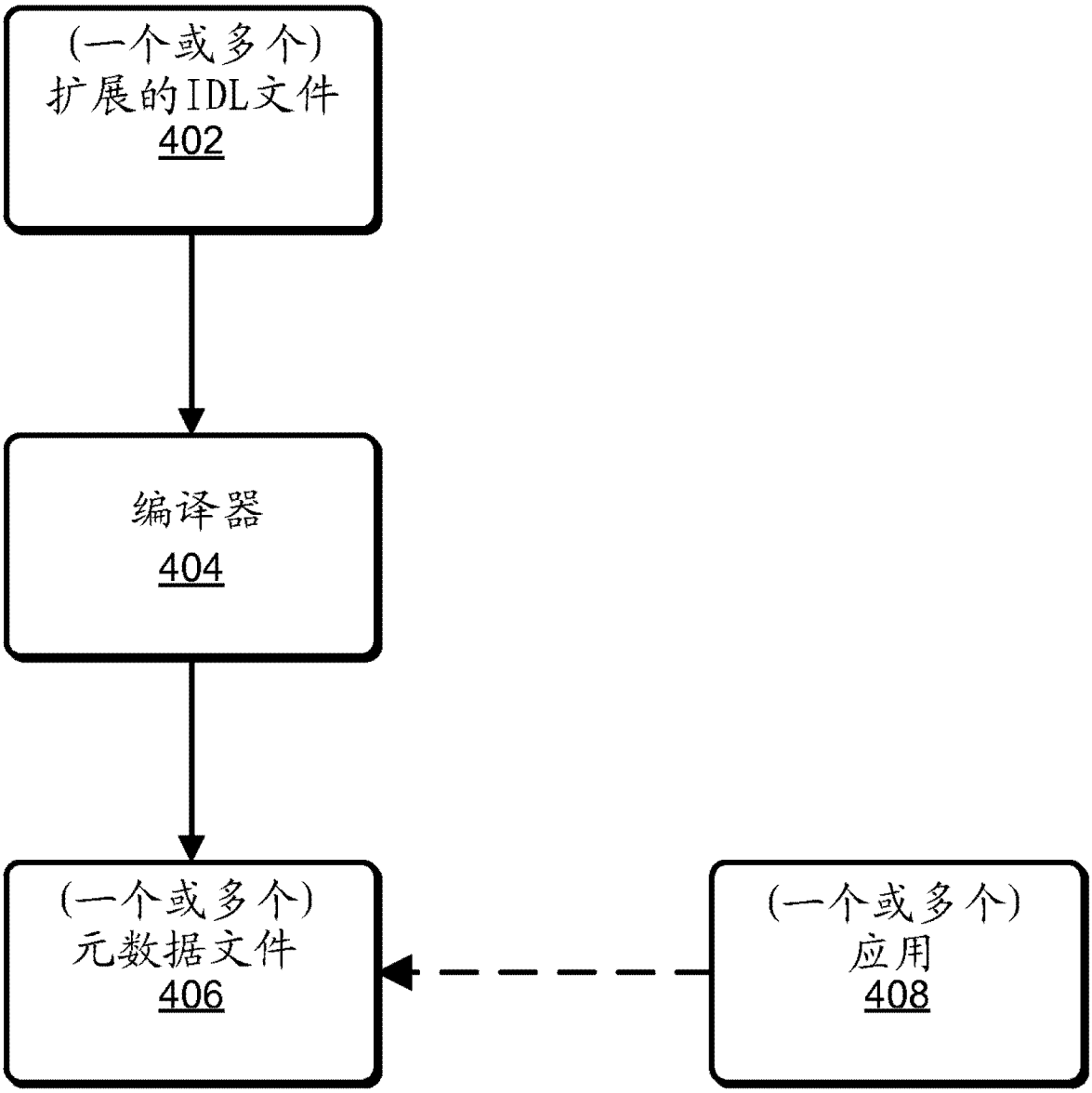


图 4

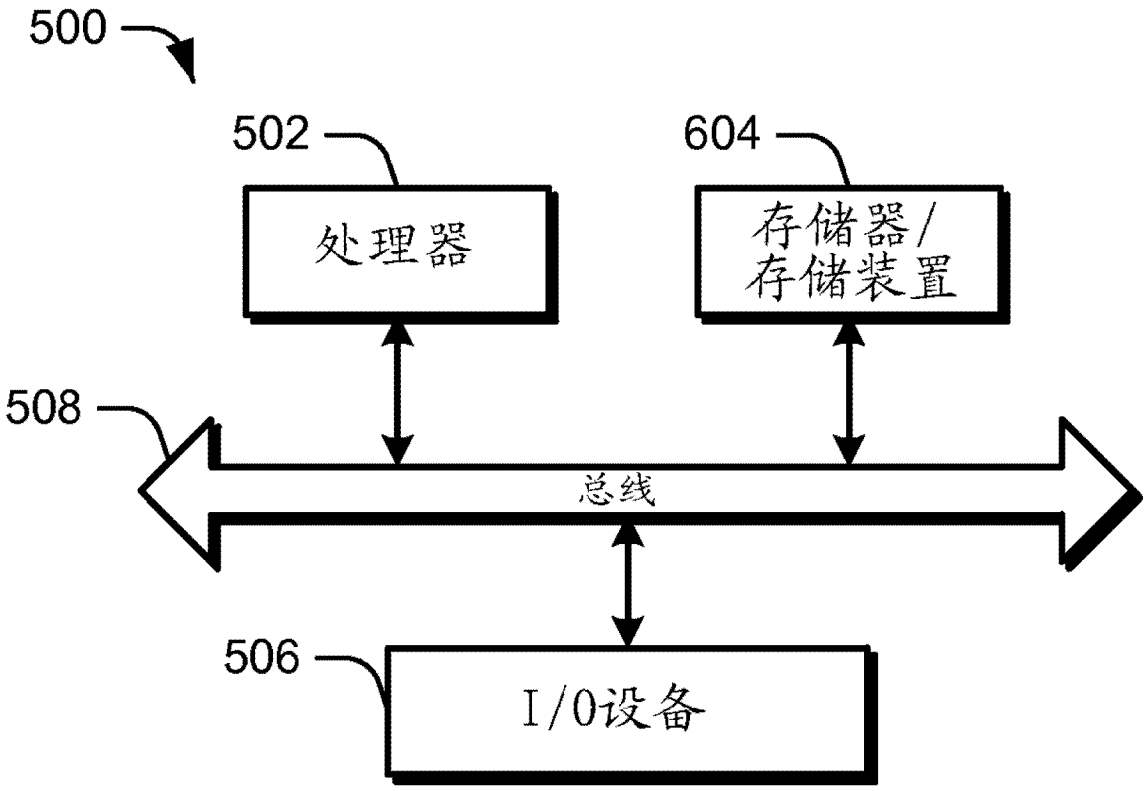


图 5